d 1-19 cit ab

1. 5,699,516, Dec. 16, 1997, Method and apparatus for implementing a in-order termination bus protocol within a data processing system; Adi Sapir, et al., 710/110; 709/209; 710/107 [IMAGE AVAILABLE]

US PAT NO: 5,699,516 [IMAGE AVAILABLE] L3: 1 of 19

ABSTRACT:
A bus protocol is provided for pipelined and/or split **transaction** buses (18,48) which have in-**order** data bus termination and which do not require data bus **arbitration**. The present invention solves the problem of matching the initial address request by a bus master (12, 13, 42) to the corresponding data response from a bus slave (14, 15, 44) when the bus (18, 48) used for master-slave communication is a split-**transaction** bus and/or a pipelined bus. Each bus master (12, 13, 42) and each bus slave (14, 15, 44) has a counter (30-33, 75-76) which is used to store a current pipe depth value (21, 51) from a central pipe counter (16, 72). A **transaction** start signal (20, 50) and a **transaction** end signal (22, 52) are used to selectively increment and decrement the counters (30-33, 75-76).

2. 5,608,879, Mar. 4, 1997, Method and apparatus for arbitrating data requests and responses thereto as separate bus transactions; Conrad C. Cooke, 710/110, 119 [IMAGE AVAILABLE]

US PAT NO: 5,608,879 [IMAGE AVAILABLE] L3: 2 of 19

ABSTRACT:
Data processing apparatus, in which at least a first and a second data handling node are connected to a common data bus, comprises an **arbitration** circuit, responsive to requests for control of the data bus from the data handling nodes, for allocating control of the data bus to selected ones of the data handling nodes for successive access periods of a predetermined length. The first data handling node comprises a processor for requesting control of the data bus for one access period in **order** to transmit a data request to the second data handling node, and the second data handling node comprises a processor, responsive to a data request received from the first data handling node, for requesting control of the data bus for a subsequent access period to transmit the requested data to the first data handling node. Data requests and the subsequent transmission of the requested data are thus treated as two separate bus **transactions**, each **transaction** taking place during an access period of a predetermined length.

3. 5,581,782, Dec. 3, 1996, Computer system with distributed bus arbitration scheme for symmetric and priority agents; Nitin V. Sarangdhar, et al., 710/119; 340/825.5; 710/108 [IMAGE AVAILABLE]

US PAT NO: 5,581,782 [IMAGE AVAILABLE] L3: 3 of 19

ABSTRACT:
A system and method for providing a high performance symmetric **arbitration** protocol that includes support for priority agents. The bus **arbitration** protocol supports two classes of bus agents: symmetric agents and priority agents. The symmetric agents support fair, distributed **arbitration** using a round-robin algorithm. Each symmetric agent has a unique Agent ID assigned at reset. The algorithm arranges the

symmetric agents in a circular **order** of priority. Each symmetric agent also maintains a [redacted] ownership state of busy or i[redacted] and a Rotating ID that reflects the symmetric agent with the lowest priority in the next **arbitration** event. On an **arbitration** event, the symmetric agent with the highest priority becomes the symmetric owner. However, the symmetric owner is not necessarily the overall bus owner (i.e., a priority agent may be the overall bus owner). The symmetric owner is allowed to take ownership of the bus and issue a **transaction** on the bus provided no other action of higher priority is preventing the use of the bus. A symmetric owner can maintain ownership without re-**arbitrating** if the **transaction** is either a bus-locked or a burst access **transaction**. The priority agent(s) has higher priority than the symmetric owner. Once the priority agent **arbitrates** for the bus, it prevents the symmetric owner from issuing any new **transactions** on the bus unless the new **transaction** is part of an ongoing bus-locked operation.

4. 5,553,268, Sep. 3, 1996, Memory operations priority scheme for microprocessors; Avigdor Willenz, et al., 711/158; 364/DIG.1; 711/151 [IMAGE AVAILABLE]

US PAT NO: 5,553,268 [IMAGE AVAILABLE] L3: 4 of 19

ABSTRACT:
A structure and a method are provided to implement a memory bus **arbiter**, in which separate priorities are provided to instruction and data reads from the main memory. In one embodiment in a microprocessor with an on-chip cache, the present invention provides an **arbiter** which yields the memory bus, in decreasing priority **order**, to an ongoing bus **transaction**, a "direct memory access" (DMA) request, an instruction read resulting from a cache miss, a pending write request, and a read request, including reference to an uncacheable portion of memory and a data cache miss.

5. 5,550,988, Aug. 27, 1996, Apparatus and method for performing error correction in a multi-processor system; Nitin V. Sarangdhar, et al., 710/113, 108, 119 [IMAGE AVAILABLE]

US PAT NO: 5,550,988 [IMAGE AVAILABLE] L3: 5 of 19

ABSTRACT:
In a multi-processor system having a first processor, a second processor, and a bus coupling the first processor to the second processor, a method for correcting an erroneous signal corresponding to the first processor while maintaining lock atomicity. When an erroneous **transaction** is detected, the first processor aborts that **transaction** and performs a retry. On the retry, an **arbitration** process **arbitrates** between the first processor and the second processor to determine which processor is granted access to the bus. If an error is detected during the **arbitration** process, an **arbitration** re-synchronization process is initiated. In the **arbitration** re-synchronization process, bus requests are de-asserted and then re-**arbitrated**. In the re-**arbitration** process, the first processor initiates its request ahead of the other processor in **order** to maintain lock atomicity.

6. 5,530,903, Jun. 25, 1996, System for reassigning a higher priority to an interrupted user by inhibiting the access of other users until the interrupted user has completed its task; Jean Calvignac, et al., 710/41; 364/230.1, 242.8, 242.9, 242.93, DIG.1; 710/42, 243, 262 [IMAGE AVAILABLE]

US PAT NO: 5,530,903 [IMAGE AVAILABLE] L3: 6 of 19

ABSTRACT:
The **arbitrating** method is based on the classification of the users

into different categori[redacted] and the assignment to all user[redacted] in a category of an identical privil[redacted] level which characterizes the[redacted] erruption capability of the users in the category. A **task** performed by a selected user in a category can only be interrupted for granting access to the resource to a user in a category having a higher privilege level. Also a normal preference level is assigned to each user within a category, which determines the selection **order** of the users in the category. The privilege level of a user category combined with the preference level of each user constitutes the priority level of the user. The access to the resource is granted to a selected user having the highest priority level. If at least one user having a privilege level higher than the privilege level of the selected user, makes a request for the resource, the **task** of the selected user is interrupted and the interrupted user is assigned an interruption preference level which is higher than the normal preference levels of all other users in the category. A new user having the highest priority level is then selected after the resource is released by the interrupting user.

7. 5,485,586, Jan. 16, 1996, Queue based arbitration using a FIFO data structure; David L. A. Brash, et al., 710/112; 340/825.5; 364/240.1, 242.6, 242.92, DIG.1; 370/462; 710/113, 240 [IMAGE AVAILABLE]

US PAT NO: 5,485,586 [IMAGE AVAILABLE] L3: 7 of 19

ABSTRACT:
A queue based **arbiter** to **arbitrate** between N devices of a computer system for access to a system bus which eliminates the need to maintain a history of bus **transactions** by queuing bus requests to track when a bus request is posted. The **arbiter** provides fair access to the bus by maintaining a queue of requests that come in from each resource in the computer system. This is accomplished by continually sampling the individual request lines of the devices to determine if a device is requesting access to the bus. Each time the **arbiter** detects a request from a device it puts an entry representative of the specific device that has requested the bus into a queue that has at least N entries. Requests are granted in the **order** that they are queued.

8. 5,457,780, Oct. 10, 1995, System for producing a video-instruction set utilizing a real-time frame differential bit map and microblock subimages; Venson M. Shaw, et al., 345/502, 501, 507, 516; 348/384, 400; 382/305 [IMAGE AVAILABLE]

US PAT NO: 5,457,780 [IMAGE AVAILABLE] L3: 8 of 19

ABSTRACT:
The present invention pertains to integrated circuit system based on novel architecture of Video-Instruction-Sec-Computing (VISC). The integrated circuit comprises a plurality of functional units to independently execute the **tasks** of remote communication, bandwidth adaptation, application control, multimedia management, and universal video encoding. The integrated circuit is also comprised of scalable formatter element connecting to the functional units which can inter-operate **arbitrary** external video formats and intelligently adapt to selective internal format depending upon the system throughput and configuration. Additionally, there is a smart memory element connecting to the functional units and scalable formatter, which can access, store, and transfer blocks of video data based on selective internal format. In the preferred embodoment, the integrated circuit is also comprised of an embedded RISC or CISC co-processor element in **order** to execute DOS, Window, NT, Macintosh, OS2 or UNIX applications In a more preferred embodiment, the integrated circuit includes a real time object oriented operation system element wherein concurrent execution of the application program and real time VISC based video instruction sets can be performed.
The present invention is designed to sustain the evolution of a plurity

generations of the VISC microprocessors. These novel VISC microprocessors can be efficiently used to perform wide range of real t[redacted] distributed video signal processing functions for applications such as interactive video, HDTV, and multimedia communications.

9. 5,367,624, Nov. 22, 1994, Interface for controlling transactions in a manufacturing execution system; Dwight H. Cooper, 345/357, 331, 332, 348, 350, 353 [IMAGE AVAILABLE]

US PAT NO: 5,367,624 [IMAGE AVAILABLE] L3: 9 of 19

ABSTRACT:
Method and apparatus for an improved interface for controlling **transactions** in a manufacturing execution system. A common display interface is displayed on a computer system display which represents a series of **transactions** for a single manufacturing process, The common display interface includes a plurality of subdisplays each separately accessible through the common display interface. Each of the plurality of subdisplays includes various information related to the step in the manufacturing process represented by the subdisplay, The common display also can access a first group of the plurality of subdisplays in a sequential **order** and perform **transactions** related to the subdisplays in a sequential **order**. The common display also can access a second group of the plurality of subdisplays at **arbitrary** time intervals, and performing **transactions** associated with the second group of the plurality of subdisplays at the **arbitrary** time intervals.

10. 5,333,296, Jul. 26, 1994, Combined queue for invalidates and return data in multiprocessor system; Gregg Bouchard, et al., 711/171; 364/239.8, 243.41, 247.7, 254.5, DIG.1; 711/117 [IMAGE AVAILABLE]

US PAT NO: 5,333,296 [IMAGE AVAILABLE] L3: 10 of 19

ABSTRACT:
A pipelined CPU executing instructions of variable length, and referencing memory using various data widths. Macroninstruction pipelining is employed (instead of microinstruction pipelining), with queuing between units of the CPU to allow flexibility in instruction execution times. A wide bandwidth is available for memory access; fetching 64-bit data blocks on each cycle. A hierarchical cache arrangement is used, increasing the likelihood of a cache hit. A writeback cache is used (instead of writethrough) and writeback is allowed to proceed even though other accesses are suppressed due to queues being full. Separate queues are provided for the return data from memory and cache invalidates, yet the **order** or bus **transactions** is maintained by a pointer arrangement. The bus protocol used by the CPU to communicate with the system bus is of the pended type, with **transactions** on the bus identified by an ID field specifying the originator, and **arbitration** for bus grant goes one simultaneously with address/data **transactions** on the bus.

11. 5,317,720, May 31, 1994, Processor system with writeback cache using writeback and non writeback transactions stored in separate queues; Rebecca L. Stamm, et al., 711/143; 364/243.41, 243.45, 254.5, DIG.1; 711/122 [IMAGE AVAILABLE]

US PAT NO: 5,317,720 [IMAGE AVAILABLE] L3: 11 of 19

ABSTRACT:
A pipelined CPU executing instructions of variable length, and referencing memory using various data widths. A writeback cache is used (instead of writethrough) in a hierarchical cache arrangement, and writeback is allowed to proceed even though other accesses are suppressed due to queues being full. Separate queues are provided for the return

data from memory and cache invalidates, yet the **order** of bus **transactions** is maintained by a pointer arrangement. The bus protocol used by the CPU to communicate with the system bus is of the pended type, with **transactions** on the bus identified by an ID field specifying the originator, and **arbitration** for bus grant goes one simultaneously with address/data **transactions** on the bus.

12. 5,191,656, Mar. 2, 1993, Method and apparatus for shared use of a multiplexed address/data signal bus by multiple bus masters; Stephen J. Forde, III, et al., 710/107; 340/825.5; 364/230, 230.4, 236.2, 238, 238.3, 239.9, 240, 240.1, 240.2, 240.5, 240.8, 240.9, 242.6, 242.92, 248.1, 259, 259.1, 259.9, 263, 271, DIG.1; 710/113, 129 [IMAGE AVAILABLE]

US PAT NO: 5,191,656 [IMAGE AVAILABLE] L3: 12 of 19

ABSTRACT:
A multiplexed address/data signal bus capable of supporting multiple bus masters includes a group of control signal lines "shared" by each bus requestor device (BRD) coupled to the bus and a group of control signal lines "replicated" into sets, one for each BRD. The bus arrangement includes a central **arbitration** unit located in a host bus interface (HBI) which controls BRD access to the bus. A selection control signal is provided to select a set of replicated control signals corresponding to a current bus master. Bus isolation units (BIUs) are provided to isolate the BRDs from the shared control signals, thereby allowing multiple BRDs to "simultaneously" utilize the signal bus in **order** to execute multiple bus **transactions** in parallel.

13. 5,191,649, Mar. 2, 1993, Multiprocessor computer system with data bus and ordered and out-of-order split data transactions; Sudarshan B. Cadambi, et al., 395/200.55; 364/228, 228.1, 228.3, 229, 232.7, 236.2, 238.4, 239.9, 240, 240.2, 240.5, 240.8, 240.9, 241.2, 242.6, 242.7, 242.92, 243, 243.4, 243.41, 244, 244.3, 248.1, 259, 259.2, 260, 260.1, 262, 262.3, 262.4, 262.9, 271.5, 280, 280.8, 281.3, 281.7, DIG.1; 710/100, 126; 711/158; 714/15 [IMAGE AVAILABLE]

US PAT NO: 5,191,649 [IMAGE AVAILABLE] L3: 13 of 19

ABSTRACT:
A method of transferring data in response to a read command in a computer system having a plurality of processors coupled to an address bus, a command bus and a data bus is described. A first processor generates and sends the read command to read a first data from a second processor. The second processor then determines with which one of (1) the first data and (2) a read response command and the first data it desires to respond to the read command. If the second processor determines to respond with the first data, then it acknowledges receipt of the read command and performs an ordered response in which the command and address buses are released and only the first data is **later** sent to the first processor via the data bus when available. If the second processor determines to respond with the read response command and the first data, then it acknowledges receipt of the read command and performs an out-of-**order** response in which the access of the command and address buses is first released and gained again by **arbitration** when the first data is determined to be available in the second processor. The second processor then gains the access of the data bus when the data bus is free of any data **transaction**. The read response command and its address and the first data are then issued to the first processor.

14. 5,155,843, Oct. 13, 1992, Error transition mode for multi-processor system; Rebecca L. Stamm, et al., 364/228, 231.8, 232.8, 239, 239.8, 239.9, 240, 240.2, 240.8, 241.9, 242.6, 242.92, 243, 243.41, 243.42, 243.44, 244, 244.3, 244.6, 244.9, 247, 247.8, 251, 251.3, 254.9, 259, 259.2, 259.5, 259.7, 259.9, 261.3, 261.4, 261.5, 261.6, 262.4, 262.7, 262.8, 262.81, 263.1, 265, 265.3, 268, 268.3, 268.5, 271.5, 280.8, 927.8,

931.4, 931.49, 931.55, 934, 934.2, 943.9, 944.92, 948, 948.34, 957, 957.6, 964, 964.2, 964[redacted], 964.32, 964.34, 964.341, 964[redacted]2, 964.343, DIG.1, DIG.2 [IMAGE AVAILABLE]

US PAT NO: 5,155,843 [IMAGE AVAILABLE] L3: 14 of 19

ABSTRACT:
A pipelined CPU executing instructions of variable length, and referencing memory using various data widths. Macroinstruction pipelining is employed (instead of microinstruction pipelining), with queueing between units of the CPU to allow flexibility in instruction execution times. A wide bandwidth is available for memory access; fetching 64-bit data blocks on each cycle. A hierarchical cache arrangement has an improved method of cache set selection, increasing the likelihood of a cache hit. A writeback cache is used (instead of writethrough) and writeback is allowed to proceed even though other accesses are suppressed due to queues being full. A branch prediction method employs a branch history table which records the taken vs. not-taken history of branch opcodes recently used, and uses an empirical algorithm to predict which way the next occurrence of this branch will go, based upon the history table. A floating point processor function is integrated on-chip, with enhanced speed due to a bypass technique; a trial mini-rounding is done on low-**order** bits of the result, and if correct, the last stage of the floating point processor can be bypassed, saving one cycle of latency. For CAL type instructions, a method for determining which registers need to be saved is executed in a minimum number of cycles, examining groups of register mask bits at one time. Internal processor registers are accessed with short (byte width) addresses instead of full physical addresses as used for memory and I/O references, but off-chip processor registers are memory-mapped and accessed by the same busses using the same controls as the memory and I/O. In a non-recoverable error detected by ECC circuits in the cache, an error transition mode is entered wherein the cache operates under limited access rules, allowing a maximum of access by the system for data blocks owned by the cache, but yet minimizing changes to the cache data so that diagnostics may be run. Separate queues are provided for the return data from memory and cache invalidates, yet the **order** or bus **transactions** is maintained by a pointer arrangement. The bus protocol used by the CPU to communicate with the system bus is of the pended type, with **transactions** on the bus identified by an ID field specifying the originator, and **arbitration** for bus grant goes one simultaneously with address/data **transactions** on the bus.

15. 5,044,619, Sep. 3, 1991, Control of pre-ordered stock; Douglas F. Sundquist, et al., 270/58.04; 414/789.5 [IMAGE AVAILABLE]

US PAT NO: 5,044,619 [IMAGE AVAILABLE] L3: 15 of 19

ABSTRACT:
A method of automatically placing ordered paper stock (covers or inserts) from sets of paper stock, the sets of paper stock comprised of an **arbitrary** number of covers or inserts, into finished copy sets. In a reproduction **job** run, and with the method of automatic recovery from a machine paper jam in which the copy sheets or paper stock in process has become misoriented comprising the steps of selecting an automatic paper stock ordering option at the operator interface, programming the required **order** of the paper stock in a finished set of reproduced documents, determining the repetitive frequency of the insert sheets, and automatically selectively inserting or discarding insert sheets in **order** to provide sets of copy sheets with uniformly placed paper stock. Also, a method for automatically recovering after an operator jam clearance by selectively purging paper stock to maintain the continuity of the ordered paper stock in the completed copy sets.

16. 5,006,982, Apr. 9, 1991, Method of increasing the bandwidth of a

packet bus by reordering reply packets; Ronald J. Ebersole, et al., 710/263; 364/229.2, 23[redacted], 242.8, 242.92, 284.1, 284.3, [redacted]G.1 [IMAGE AVAILABLE]

US PAT NO: 5,006,982 [IMAGE AVAILABLE] L3: 16 of 19

ABSTRACT:
A data processor bus in which information is transferred between agents attached to the bus by issuing request packets that request data from an agent on the bus and reply packets that return data requested by a request packet. A control method mixes request-and-reply packets on the bus by determining the use of a next-bus cycle using **arbitration**, reply deferral, and specification lines and the state of a grant queue and a pipe queue in accordance with a specified protocol. A request is forced to take the next available bus cycle upon the condition that there is an agent identified in the great queue and the pipeline queue is not full. A reply packet is forced to take the next available bus cycle upon the condition that the pipeline queue is full. A reply packet is forced to take the next available bus cycle upon the condition that the grant queue is empty and the pipeline queue is not empty. Giving requests precedence over replies to allows the pipeline to be kept as full as possible. A replying agent assigned to the highest priority slot 1 in the pipeline queue is allowed to defer its own slot in the pipeline queue until a **later** time to thereby permit a **transaction** in Slot 2 of the pipeline queue to be completed before the one ahead of it.

17. 4,920,485, Apr. 24, 1990, Method and apparatus for arbitration and serialization in a multiprocessor system; Ali Vahidsafa, 710/242; 364/228, 229.1, 230, 230.3, 242.6, 242.7, 262.4, 263.2, 271.5, DIG.1 [IMAGE AVAILABLE]

US PAT NO: 4,920,485 [IMAGE AVAILABLE] L3: 17 of 19

ABSTRACT:
A method and apparatus for providing **arbitration** between and serialization of plural processors in a multiprocessor system comprising, in each processor, a delay network, a priority circuit, a REQUEST generator, an **ORDER** generator, a serialization program, an ACK generator and an ACK receiver. In operation, the delay network insures that simultaneously generated REQUESTS received from plural processors are received by the priority circuit at the same time. A processor awarded priority issues an **ORDER** to the other processors and thereafter drops its REQUEST to allow an award of priority to another processor. An ACK is received by the **ORDER** issuing processor from each processor when it executes the **ORDER**. The **ORDER** issuing processor then completes the **task** which gave rise to the **ORDER**. To conserve processing time, priority awards may be made before previously issued ORDERS are completed. Alternatively, REQUEST issuing processors can simply hold their REQUEST and thereby prevent interruption of instructions or groups of instructions.

18. 4,358,278, Nov. 9, 1982, Learning and matching apparatus and method; Adolph E. Goldfarb, 434/337, 169, 227 [IMAGE AVAILABLE]

US PAT NO: 4,358,278 [IMAGE AVAILABLE] L3: 18 of 19

ABSTRACT:
An electronic learning apparatus and methodology is disclosed wherein answer codes to a plurality of questions or **tasks** are disposed on the periphery of a disc. The answer codes are photo-optically read from the periphery of the disc as the disc is rotated with the answer thus read being stored in a register. The user inputs a student's answer through a keyboard into a second register. The answer code as read from the disc is compared to the keyed answer from the user to give an indication of correctness or error. The invention also includes a method for teaching

arithmetic operations wherein the user must input the nature of the arithmetic operation as well as the result in **order** to complete the questions. The invention is also characterized in that the keyboard is adapted to be combined with a plurality of keyboard overlays whereby an **arbitrary** number of associations may be made between the fixed number of keys and an indefinite number of response types.

19. 4,326,098, Apr. 20, 1982, High security system for electronic signature verification; Willard G. Bouricius, et al., 380/25; 235/380; 340/825.3; 380/29, 37, 45, 49 [IMAGE AVAILABLE]

US PAT NO: 4,326,098 [IMAGE AVAILABLE] L3: 19 of 19

ABSTRACT:
The system provides both electronic signature and message verification with a minimum of excess coding information on an instantaneous basis and is easily restartable in a store and forward environment. The system is based on the concept of a vault or central authority. The vault is in essence a physically secured Authenticator designed as a hardware automation which is not under control of any operating system. The system is a terminal based network wherein all terminals or users may communicate directly or through a central CPU. All secure electronic signature verification **transactions** must be transacted through the central facility which includes said vault. The vault and all terminals include an identical key-controlled block-cipher cryptographic facility wherein each user at a terminal has access only to his own key and wherein the vault has access to all user keys. At the end of a **transaction**, a user A (originator) and a user B (receiver) each have uniquely encrypted messages which can be utilized in **later arbitration** proceedings wherein user A cannot **later** deny having sent a message or its contents and similarly user B cannot deny having received the message or its specific content. The vault provides facilities for effective legal **arbitration** and is also simple to operate in such a n-to-n network without using more than one key per person.